RHEINISCH-
WESTFÄLISCHE
TECHNISCHE
HOCHSCHULE
AACHEN

LEHRSTUHL
INFORMATIK XI
SOFTWARE FÜR
EINGEBETTETE
SYSTEME

Professor Dr.-Ing. Stefan Kowalewski
Dipl.-Inform. Bastian Schlich

Aachen, 17. Januar 2005
SWS: V2/Ü2

Systemprogrammierung

WS 2004/2005

Übungsblatt 12

Abgabe der Lösungen: ab 24. Januar 2005 in den Übungen

Aufgabe 1 (5 Punkte): Hauptspeicherverwaltung

Gegeben ist ein Betriebssystem mit einem virtuellen Speicher auf der Basis des Pagingverfahrens. Betrachten Sie einen Prozess, der die folgende Sequenz von Speicherzugriffen (Seitennummern) erzeugt:

3,5,0,3,1,2,3,5,4,1,5,3

Dem Prozess stehen 4 (physikalisch vorhandene) Seitenrahmen zur Verfügung, die anfangs nicht belegt sind.

Geben Sie für alle Teilaufgaben die Belegung der Seitenrahmen in einer Tabelle an. Legen Sie für jeden Seitenrahmen (0-3) eine Zeile, und für jeden Speicherzugriff eine Spalte an. Tragen Sie für jeden Speicherzugriff in die Tabelle ein, welche Seite welchem Seitenrahmen zugeordnet ist. Beachten Sie, dass eine Seite den Seitenrahmen nicht wechselt, während die Seite eingelagert ist. Markieren Sie jede Spalte in der ein Seitenfehler auftritt. Die erste Einlagerung in die noch leeren Seitenrahmen zählt als Seitenfehler.

a) Bestimmen Sie die Anzahl der Seitenfehler bei Benutzung der FIFO (First-In-First-Out) Ersetzungsstrategie

b) Führen Sie dasselbe für die LRU (Least-Recently-Used) Strategie durch.

c) Beschreiben Sie, wie die optimale Strategie (OPT) für eine minimale anzahl von Seitenfehlern arbeiten müsste. Warum ist dies in der Praxis meist nicht möglich?

d) Bestimmen Sie die minimale Anzahl von Seitenfehlern bei Verwendung der OPT Strategie.

e) Diskutieren Sie kurz Probleme, die im Zusammenhang mit Segmentierung und Paging auftreten.

Lösung 1

a) Es treten 9 Seitenfehler auf:

	3	5	0	3	1	2	3	5	4	1	5	3
0	3	3	3	3	3	2	2	2	2	1	1	1
1		5	5	5	5	5	3	3	3	3	3	3
2			0	0	0	0	0	5	5	5	5	5
3					1	1	1	1	4	4	4	4
P	x	x	x			x	x	x	x	x	x	

b) Es treten 8 Seitenfehler auf:

	3	5	0	3	1	2	3	5	4	1	5	3
0	3	3	3	3	3	3	3	3	3	3	3	3
1		5	5	5	5	2	2	2	2	1	1	1
2			0	0	0	0	0	5	5	5	5	5
3					1	1	1	1	4	4	4	4
P	x	x	x		x	x		x	x	x		

c) Es wird die in der Zukunft am längsten nicht benötigte Seite ersetzt. Normalerweise sind die Zugriffe nicht im voraus bekannt.

d) Es treten 6 Seitenfehler auf:

	3	5	0	3	1	2	3	5	4	1	5	3
0	3	3	3	3	3	3	3	3	3	3	3	3
1		5	5	5	5	5	5	5	5	5	5	5
2			0	0	0	2	2	2	4	4	4	4
3					1	1	1	1	1	1	1	1
P	x	x	x		x	x			x			

e) Interne bzw. externe Segmentierung.

Aufgabe 2 (6 Punkte): UNIX-Scheduling

In dieser Aufgabe wird das Scheduling von BSD Unix genauer betrachtet. Es handelt sich dabei um eine Zeitscheibenstrategie mit dynamischer Prioritätenvergabe. Unix vergibt für seine Prozesse Prioritäten von 0 bis 127 (0 ist die höchste Priorität), die in 32 Warteschlangen verwaltet werden. Ein Prozess mit Priorität PRIO wird in die Schlange PRIO/4 eingeordnet. Alle Prozesse einer Prioritätsklasse befinden sich in einer Warteschlange, die nach einer Round-Robin Zeitscheibenstrategie abgearbeitet wird. Zunächst wird allen Prozessen der höchsten Priorität die CPU zugeteilt, bis die Warteschlange leer ist. Dann kommen die Prozesse mit der nächstniedrigeren Priorität zum Zuge.

Die Prioritäten werden fortlaufend neu berechnet (multilevel-feedback-queue). Prozesse, die in einem gewissen Zeitabschnitt viel Rechenzeit verbraucht haben, werden in ihrer Priorität erniedrigt, und Prozesse, die lange gewartet haben, erhalten eine höhere Priorität ('short-term-scheduling'). Die Prioritäten werden folgendermaßen berechnet:

1. $p_{cpu} = \left\lceil \frac{2 \cdot load}{2 \cdot load + 1} \cdot p_{cpu} + p_{nice} \right\rceil$,
   wobei *load* eine Abschätzung der CPU-Auslastung ist.

2. $u_{prio} = \left\lceil \text{USER\_PRIO} + \frac{p_{cpu}}{4} + 2 \cdot p_{nice} \right\rceil$,
   wobei p_{cpu} die Prozessornutzung des rechnenden Prozesses ist und alle 10ms um 1 inkrementiert wird. p_{nice} ist ein vom Benutzer bestimmter Gewichtungsfaktor ($-20 \leq p_{nice} \leq 20$), und USER_PRIO ist die Priorität, die dem Prozess beim Start zugeteilt worden ist.

a) Gegeben seien die folgenden Prozesse:

Prozess	Bedienzeit	Priorität
1	10	3
2	1	1
3	2	3
4	1	4
5	5	2

Geben Sie die Abarbeitungsfolge der Prozesse, die sich bei Anwendung der oben erläuterten Strategie des 'short-term-schedulings' ergibt, in einer Tabelle mit allen Werten an. Nehmen Sie zur Vereinfachung an, dass ein Zeitquantum den Wert 1 hat, p_{cpu} (des rechnenden Prozesses!) in jedem Zeitquantum um 8 erhöht wird, p_{nice} den Wert 0 hat und *load* gleich 1 ist. Die Priorität aller Prozesse soll nach jedem vierten Quantum neu berechnet werden.

b) Visualisieren Sie die CPU-Nutzung durch ein Gantt-Chart.

c) Welche Beweggründe stehen hinter dem 'short-term-scheduling'?

Lösung 2

a) Berechnung des SchedulesMit den angegebenen Werten ergeben sich folgende Formeln:

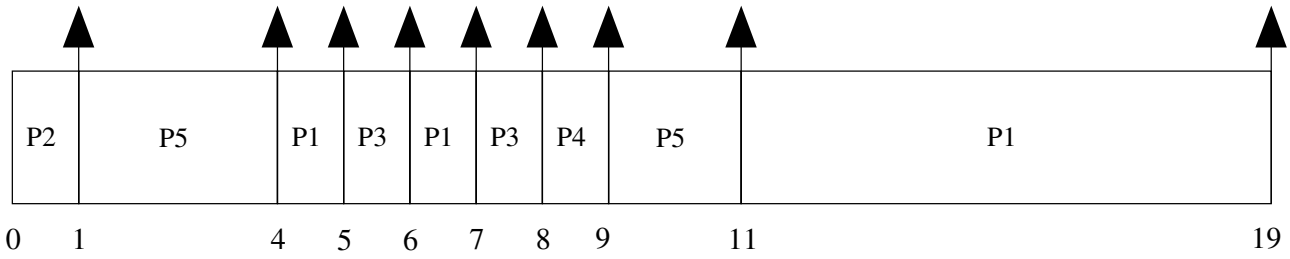
$$u_{prio} = \text{USER\_PRIO} + \frac{p_{cpu}}{4} \quad (1)$$

$$p_{cpu} = \frac{2}{3} \cdot p_{cpu} \quad (2)$$

Weiterhin seien t_b die restliche Bedienzeit und p die aktuelle Priorität. Die Abarbeitungsreihenfolge des 'short-term-scheduling' kann folgender Tabelle entnommen werden:

	P_1			P_2			P_3			P_4			P_5		
t	t_b	p	p_{cpu}	t_b	p	p_{cpu}	t_b	p	p_{cpu}	t_b	p	p_{cpu}	t_b	p	p_{cpu}
	10	3	0	1	1	0	2	3	0	1	4	0	5	2	0
1				0	1	8									
2													4	2	8
3													3	2	16
4													2	2	24
	10	3	0	-	-	-	2	3	0	1	4	0	2	6	16
5	9	3	8												
6							1	3	8						
7	8	3	16												
8							0	3	16						
	8	6	11	-	-	-	-	-	-	1	4	0	2	5	11
9										0	4	8			
10													1	5	19
11													0	5	27
12	7	6	19												
	7	6	13	-	-	-	-	-	-	-	-	-	-	-	-
13	6	6	21												
14	5	6	29												
15	4	6	37												
16	3	6	45												
	3	10	30	-	-	-	-	-	-	-	-	-	-	-	-
17	2	10	38												
18	1	10	46												
19	0	10	54												

b) Gantt-Chart



c) Das 'short-term-scheduling' unterstützt das interaktive Arbeiten am Rechner. Die Priorität von Prozessen, die lange auf die Ausführung von E/A-Operationen warten (z.B. die Shell), wird erhöht. Dagegen werden die Prioritäten von Prozessen, die wesentliche CPU-Zeiten auf sich vereinigen, verringert. Dies bedeutet zusammenfassend, daß 'interaktive' Prozesse den 'Batch'-Prozessen vorgezogen werden.

Aufgabe 3 (2+2+2+2=8 Punkte): Dateisysteme

In der Vorlesung haben Sie Konzepte von Dateisystemen kennen gelernt. Diskutieren Sie folgende Fragen.

a) Betrachten Sie ein Dateisystem, in dem Dateien gelöscht werden können und der deren Platz als frei deklariert wird obwohl Links zu diesen Dateien existieren. Welche Probleme können auftreten, wenn neue Dateien mit gleichen absoluten Namen oder in den selben Bereichen des Dateisystems angelegt werden?

b) Manche Betriebssysteme löschen automatisch alle Benutzerdateien, wenn der Nutzer sich aus dem System abmeldet oder wenn eine Anwendung terminiert. Um die Daten zu behalten, muss der Nutzer dies explizit sagen. Im Gegensatz dazu gibt es Betriebssysteme, die alle Nutzerdateien speichern bis der Nutzer selber sie löscht. Welche relative Vorteile bzw. Nachteile haben beide Strategien?

c) Geben Sie ein Beispiel für eine Anwendung, die entweder

   i) sequentiell oder

   ii) zufällig

   auf die Daten im Dateisystem zugreift:

d) Welche Probleme können auftreten, falls man zulassen würde, dass ein Dateisystem simultan an mehreren Einhängepunkten (mountpoints) eines Systems eingehängt werden darf?

Lösung 3

a) Es kann zu Dateninkonsistenzen kommen. Wenn Programme versuchen über diese Links Dateien zu lesen, dann bekommen Sie unter diesen Umständen falsche Daten.

b) Löscht das Betriebssystem alle Dateien des Nutzers so kann es nicht passieren, dass der Nutzer mit eigenen Daten den gesamten freien Platz verbraucht. Andererseits, wenn der Nutzer vergisst dem System mitzuteilen die Daten zu bewahren, kann es zur ungewollten Datenverlust kommen.

c) Ein Musikplayer greift sequentiell auf die Musikdaten zu. Verschlüsselungsprogramme können (pseudo-)zufällig Daten auf dem Dateisystem lesen.

d) Wenn beispielsweise zwei Programme auf das gleiche Dateisystem und in das gleiche Verzeichnis schreiben würden, dann würde es offensichtlich zu Konflikten kommen, da für das Betriebssystem zwei verschiedene Verzeichnisse existieren. Zur Vermeidung wäre eine aufwendige Verwaltung der geöffneten Dateien notwendig.